TP n°2 (Encore des Threads)
Master TSD1

Exercice n°1 (Un problème d'accès concurrent)

soit la classe Compte et la classe Operation suivantes

```
public class Compte {
  private int solde = 0;

  public void ajouter(int somme) {
    solde += somme;
    System.out.print(" ajoute " + somme);
  }

  public void retirer(int somme) {
    solde -= somme;
    System.out.print(" retire " + somme);
  }

  public void operationNulle(int somme) {
    solde += somme;
    System.out.print(" ajoute " + somme);
    solde -= somme;
    System.out.print(" retire " + somme);
  }

  public int getSolde() {
    return solde;
  }
}

public class Operation extends Thread {
  private Compte compte;

  public Operation(String nom, Compte compte) {
    super(nom);
    this.compte = compte;
  }

   public void run() {
    while (true) {
      int i = (int) (Math.random() * 10000);
      String nom = getName();
      System.out.print(nom);
//        compte.ajouter(i);
//        compte.retirer(i);
      compte.operationNulle(i);
      int solde = compte.getSolde();
      System.out.print(nom);
      if (solde != 0) {
        System.out.println(nom + ":**solde=" + solde);
        System.exit(1);
      }
    }
  }

  public static void main(String[] args) {
    Compte compte = new Compte();
```

```
    for (int i = 0; i < 20; i++) {
      Operation operation = new Operation("" + (char)('A' + i), compte);
      operation.start();
    }
  }
}
```

1) Examinez le code et faites exécuter la classe Opération. Constatez le problème : opération effectue des opérations qui devraient laisser le sode du compte inchangé, et pourtant, après un moment, le solde ne reste pas à 0. Expliquez.
2) Modifiez le code pour empêcher ce problème.
3) Dans le code de Operation, remplacez l'opération nulle par 2 opérations ajouter et retirer qui devraient elles aussi laisser le solde du compte à 0 (elles sont en commentaire dans le code). Lancez l'exécution et constatez le problème. Modifiez le code pour que ça marche.

Exercice n°2 (Tri Parallèle)

Soit la classe Trieur suivante qui permet d'effectuer un tri d'un tableau en le divisant en sous tableaux de dimensions 2 par des appels récursifs et en fusionnant les tableaux triés.

```
/**
 * Tri d'un tableau d'entiers
 * Version mono-thread
 */
public class Trieur {

  private int[] t;

  private Trieur(int[] t) {
    this.t = t;
  }

  /**
   * Trie un tableau d'entiers par ordre croissant
   * @param t tableau à trier
   */
  public static void trier(int[] t) {
    Trieur tableau = new Trieur(t);
    tableau.trier(0, t.length - 1);
  }

  /**
   * Trie une tranche de t
   * @param debut indice du début de la partie à trier
   * @param debut indice de la fin de la partie à trier
   */
  private void trier(int debut, int fin) {
    if (fin - debut < 2) {
      if (t[debut] > t[fin]) {
        echanger(debut, fin);
      }
```

```
  }
  else {
    int milieu = debut + (fin - debut) / 2;
    trier(debut, milieu);
    trier(milieu + 1, fin);
    triFusion(debut, fin);
  }
}

/**
 * Echanger t[i] et t[j]
 */
private void echanger(int i, int j) {
  int valeur = t[i];
  t[i] = t[j];
  t[j] = valeur;
}

/**
 * Fusionne 2 tranches déjà triées du tableau t.
 *   - 1ère tranche : de debut à milieu
 *   - 2ème tranche : de milieu + 1 à fin
 * @param milieu indique le dernier indice de la 1ère tranche
 */
private void triFusion(int debut, int fin) {
  // tableau où va aller la fusion
  int[] tFusion = new int[fin - debut + 1];
  int milieu = (debut + fin) / 2;
  // Indices des éléments à comparer
  int i1 = debut,
      i2 = milieu + 1;
  // indice de la prochaine case du tableau tFusion à remplir
  int iFusion = 0;
  while (i1 <= milieu && i2 <= fin) {
    if (t[i1] < t[i2]) {
      tFusion[iFusion++] = t[i1++];
    }
    else {
      tFusion[iFusion++] = t[i2++];
    }
  }
  if (i1 > milieu) {
    // la 1ère tranche est épuisée
    for (int i = i2; i <= fin; ) {
      tFusion[iFusion++] = t[i++];
    }
  }
  else {
    // la 2ème tranche est épuisée
    for (int i = i1; i <= milieu; ) {
      tFusion[iFusion++] = t[i++];
    }
  }
  // Copie tFusion dans t
  for (int i = 0, j = debut; i <= fin - debut; ) {
    t[j++] = tFusion[i++];
  }
}

public static void main(String[] args) {
  int[] t = {5, 8, 3, 2, 7, 10, 1};
  Trieur.trier(t);
  for (int i = 0; i <t.length; i++) {
    System.out.print(t[i] + " ; ");
```

```
    }
    System.out.println();
  }
}
```

Vous pouvez exécuter le programme en mode monotâche pour tester qu'il fonctionne correctement.

1) Proposez une solution multi-tâches de cet exercice en utilisant les threads. Trouvez vous un problème lors de l'affichage.

2) Corrigez le problème en synchronisant l'affichage avec la fin d'exécution des Threads.